

Project Software Engineering

(DLMCSPSE01_CF)

Inhalt

- Abstract 2
- Project plan 2
- Risks..... 3
 - Business Risks 3
 - Technical Risks..... 4
 - Feature regression Risk 4
 - GPT pricing risk..... 4
 - Security risks..... 4
- Software development methodology 7
 - Kanban Board 7
 - Epic Board..... 7
 - Milestone View..... 8
 - Roles 8
- Requirements 9
 - Non-functional..... 9
 - Functional 10
- Glossary 12
- System Design 13

Abstract

The goal of this project is to develop an AI-based nutritional assistant. This assistant should help users improve their diet and overall health.

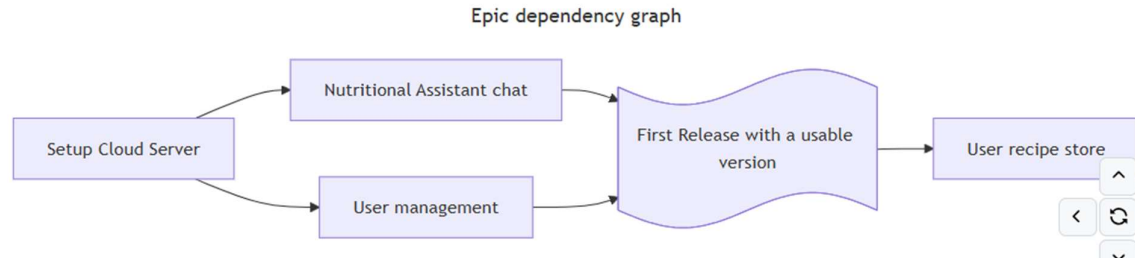
The assistant targets people who struggle with healthy diets because they lack the right information, or people who already have the basic knowledge but need special advice in their daily lives. Since this is a digital offer, the early adopters are digitally savvy and probably young people.

Project plan

The first milestone: the app should only support basic chat functionalities between the user and the digital assistant. To demonstrate a first more advanced feature, the first assistant tool will be implemented. The first tool will allow the assistant to store recipes in the database. With this, the first milestone will be implemented with those four epics and the scope of this assignment.

[EPIC]: Server setup	0 / 0	epic
#53 - by djochim was closed 2 weeks ago · Duplicate · v1		
[EPIC]: User management	0 / 4	epic
#47 - djochim opened 2 weeks ago · v1		
[EPIC]: Nutritional assistant chat	0 / 2	epic
#40 - djochim opened 2 weeks ago · v1		
[EPIC]: User recipe store	0 / 2	epic
#39 - djochim opened 2 weeks ago · v1		
[EPIC]: Setup cloud server	6 / 6	epic
#1 - by djochim was closed 6 minutes ago · v1		

The following chart shows the dependencies of the epics to each other. It also shows the first usable version, that brings the first benefits to the user.



Risks

As every software project also this projects has to mitigate certain risks which can be grouped in business related risks, technological risks and security risks.

Business Risks

Data Privacy Risk

The solution experimented in this project is a 100% digital offer and the privacy concerns with those might scare off users. Especially because the assistant needs personal information about the user, to be able to support them with personalized advice.

Mitigation:

As a mitigation, data privacy should be a top priority. Users should be aware of every personal data, that is shared with the user. The User should also be able to view all data which the app collects and should be able to adapt / delete those.

In general the App should emphasize the importance of user privacy by providing good user assistance and the right data privacy features.

Product Market Risk

The problem of a healthy diet is huge and so are the solutions for it. It could be that the developed app is not a good solution to the users dietary problems. This would lead to a high user churn.

Mitigation:

The first mitigation would be a user survey, but this will not be done as part of this assignment. Instead, the first implementation will rely on the Problems of the Project member. After the first MVP is implemented, the app can be iterated based on customer feedback. This way the app can achieve product market fit over time.

AI Legal Risk

As outlined before, the goal is to use an AI to answer the questions to the user. The output of those generative models can only be controlled in a limited fashion. The risk is, that the AI provides some negative advice to a user.

Mitigation:

Implement clear user consent mechanism and make transparent, how generative AI works and that the recommendation's must be considered carefully and that those are no medical advices. In addition, the context definition of the assistant itself should contain guidelines to the AI, which recommendations are ok and what the AI should not do. In addition broadly adapted models should be used which are already shipped / trained with guidelines and content restrictions.

AI Wrapper Risk

An app, based on a GPT model enables problem-solving of complex problems in a limited amount of time. The risk is that at some point a generic solution also based on GPT (e.g. Chat-GPT) can do all this on their one.

Mitigation:

Always provide additional features in the specific Problem space, that other solutions are not supporting. In the first milestone the more Problem focused App visualization and the feature to store generated recipes for later use are two differentiators. In addition, the Assistant is already configured with an optimized Prompt, which a normal User will maybe not be able to write.

Technical Risks

Feature regression Risk

The software solution will be developed in multiple iterations and with small feature enhancements. It is important to prevent feature regressions, where some functionality is no-longer working as expected.

Mitigation:

Write automated tests and use CI/CD pipelines to deploy changes only when the tests are passed. The tests should be separated in low level unit tests covering critical functionalities (70% line coverage) and system tests, which test the backend service and backend client as a whole.

GPT pricing risk

The app heavily depends on GPT models which already have a certain price. If a model increases the price significantly, this impacts the operational cost of the application.

Mitigation:

In Milestone one only the GPT model of OpenAI will be connected. It must be connected using an Adapter, that will allow an easy switch to a different model at a later point in time. That way the APP could switch the Model with low effort, if the used model changes the price or another model gets cheaper or leads to better results.

Security risks

Since the app is handling critical user data, security plays an important role. All risks must be prioritized according to priority.

Unauthorized Access to Sensitive Health Data

If the User Data is not properly secured, attackers can get unauthorized access to the sensitive personal and health-related information. This can result from database misconfigurations, weak encryption or inadequate access controls.

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	High
Loss of Integrity	Possible	Moderate	Moderate
Loss of Availability	Possible	Significant	High
		Overall Risk	High

Mitigations:

1. Encrypt Data at rest and in transit
2. Strict access controls
3. Secure storage using zero trust model (avoid database exposure)
4. Apply rate limiting

Data leakage from AI Models

If the Users Health Data is shared with the AI, this can imply critical data protection issues. Personal health data is very sensitive and special mechanisms must be applied.

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Likely	Moderate	Moderate
Loss of Integrity	Unlikely	Moderate	Moderate
Loss of Availability	Very Unlikely	Negligible	Low

	Overall Risk	Moderate
--	---------------------	----------

Mitigations:

1. Do not share sensitive health data with the assistant
2. Inform the user, that no personal health data should be shared
3. Instruct the AI model to not ask for personal health information
4. Use model with Health Insurance Portability and Accountability Act (HIPAA) support
 - a. Might be important for later features
 - b. Those models also allow zero retention policies

Weak or compromised Authentication

Weak authentication mechanisms or poor session management could allow unauthorized access.

<i>Thread Event</i>	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	High
Loss of Integrity	Unlikely	Moderate	Moderate
Loss of Availability	Possible	Moderate	Moderate
		Overall Risk	High

Mitigations:

1. Strict password policy
2. Support multifactor authentication
3. Use short sessions and use refresh tokens for session security
4. Secure App access with biometric encryption

API Abuse and Unauthorized Data Access

Attacker can use not properly secured APIs to exploit them and to extract user data or perform unintended actions

<i>Thread Event</i>	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Moderate	Moderate
Loss of Integrity	Unlikely	Minor	Low
Loss of Availability	Possible	Moderate	Moderate
		Overall Risk	Moderate

Mitigations:

1. All APIs secured using JWT tokens
2. Rate limiting and request throttling
3. Use a web application firewall (WAF) to filter and block malicious requests
4. Strong input validation to prevent injection attacks
5. Use secure TLS versions
6. Auditlog suspicious activity

Insecure Communication Between Components

The communication between components uses Protobuf and HTTP protocols. If these communications are not secured, attackers can intercept and or manipulate data.

Ease of Attack: Medium

<i>Thread Event</i>	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	Moderate

<i>Loss of Integrity</i>	Possible	Moderate	Moderate
<i>Loss of Availability</i>	Possible	Moderate	Moderate
		Overall Risk	Moderate

Mitigations:

1. Use TLS 1.3 where possible and do not allow versions lower than 1.2
2. Implement mTLS between system components
3. Apply HSTS headers to prevent protocol downgrade attacks

Software development methodology

This project will use the Kanban software development methodology. Kanban is very lightweight which makes it suitable for this small one person project. It does not require complex planning and does not force some cycles, but still allows to be flexible and to change priorities.

Kanban Board

The core focus in Kanban is the Kanban-Board. This project will be managed via Github-Projects and the Kanban board has the six columns *Backlog*, *Refined*, *Ready*, *In progress*, *In review* and *Done*.

The Backlog contains a list of all tasks for a bigger timeframe. It has a very big limit of 20, because in this area all items are collected for the future. Items in the Backlog are only roughly described and are not estimated and not prioritized.

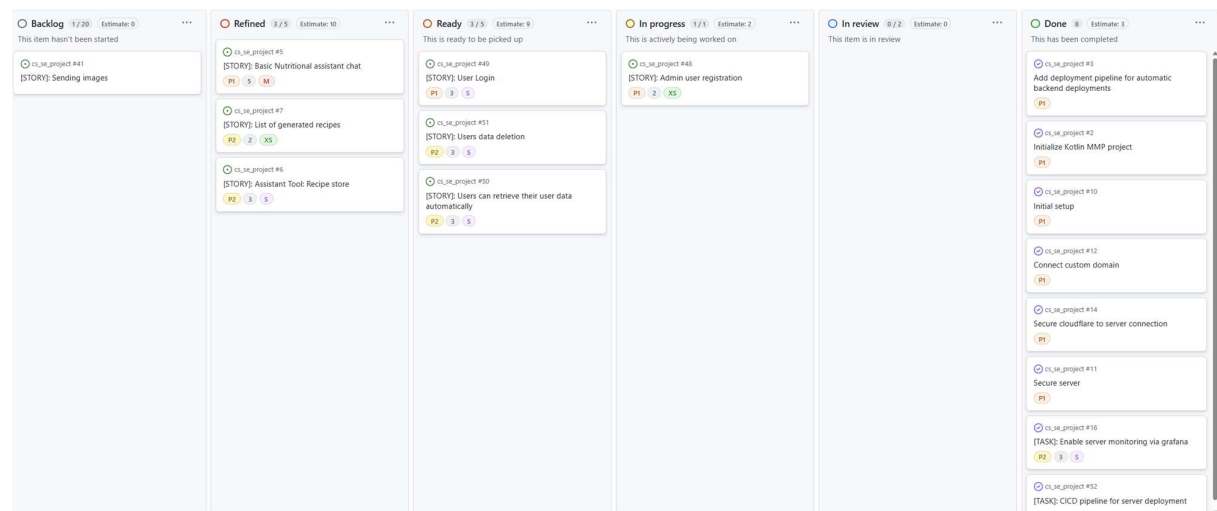
Items which are well described and which will be tackled in the next weeks, are in the Refined column. The column is limited to five prioritized and estimated items, which are ready for development as soon as all dependent requirements are fulfilled.

Once the refined items are completely unblocked, they can move in the ready column. These items could be started at any point in time. This column is also limited to five items and combined with the five refined items, they show the next up to ten items that will be done.

Started items are moved to in progress. Since this is a one-person development project, the progress column is restricted to one item. With the multitasking will be prevented.

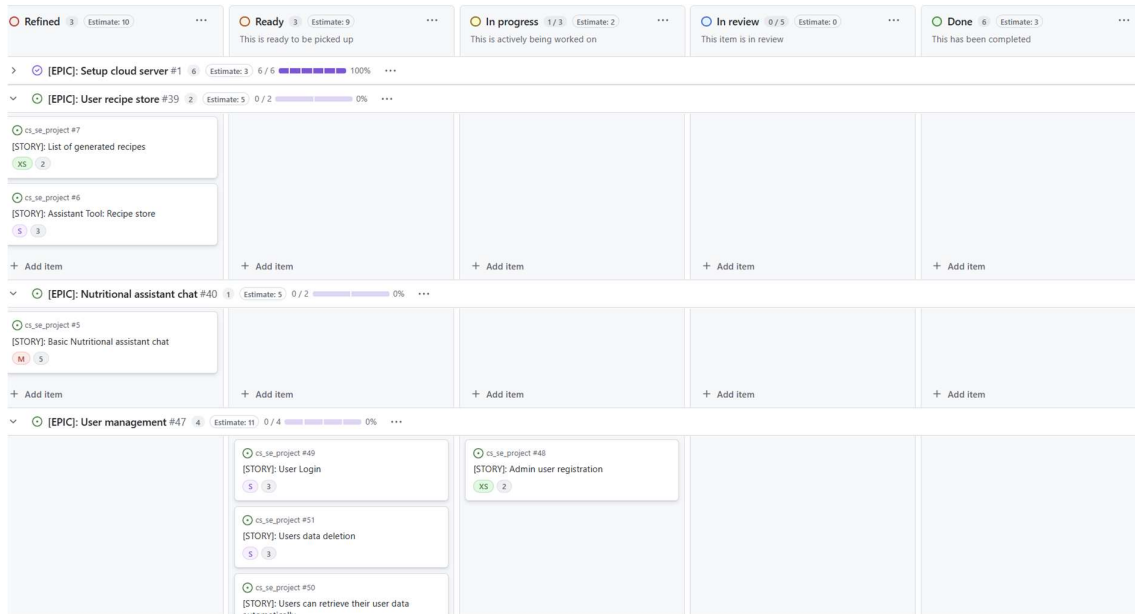
Once a PR is opened, items are automatically moved to *In review*. There are no second eyes, that could review the code, but there are automated checks, which prevent security breaches and software regressions.

After a PR is merged, items are moved to *Done*. The *Done* column has no limit configured.



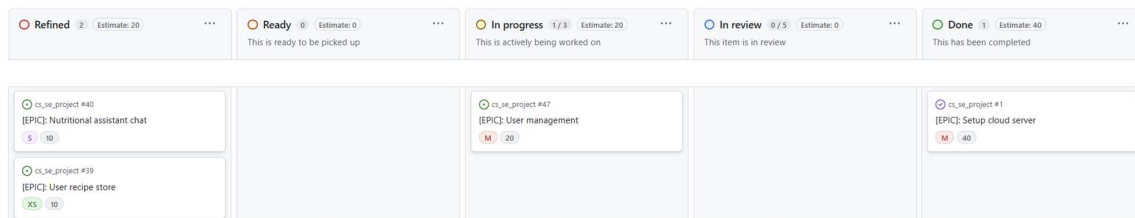
Epic Board

The epic board shows a high level overview, how far the epics have progressed and which tasks/stories there are left to complete an epic.



Milestone View

The milestone view shows the status of the overall milestone and which epics are open until the milestone is completed.



Roles

The Kanban project requires multiple roles which will all be filled with the one person in the exceptional project.

The first role is the Product Owner, who is responsible for creating the backlog and maintaining priorities based on user requirements. The Epics and User stories which the product owner defines, are further refined by the development team. This team consist of architects, developers and DevOps Engineers. The Architect is responsible for the system design and application architecture. The DevOps-Engineers create a continuous integration and continuous development pipeline. The pipeline ensures that new features are release often and only if the quality stages are passed.

Developers are responsible for the implementation of the tasks and stories. They are also responsible to write good/automated tests that ensure that new releases don't introduce regressions and that the manual testing effort is low.

Requirements

The requirements can be grouped into non-functional and functional requirements. The non-functional requirements are required to deliver a good software solutions with high quality.

Non-functional

These non-functional requirements can be separated into quality requirements and technical requirements. The quality requirements ensure, that the software works as expected and can be used by the user. For this project the main concentration will be on software security, because the data which is handled might be critical, and it is important for the user to trust this new assistant.

The risks section already outlines the security risks and those must be mitigated by implementing security measures.

To ensure the confidentiality of the system, every communication is encrypted with TLS and communication over the public network is also using mTLS to prevent man-in-the-middle attacks of a malicious actor.

In order to be resistance under attack, the risk of Denial of service attacks must be minimized by leveraging third party prevention services. In addition, all libraries must be up-to-date and security patches applied quickly. Do ensure this, a concept of automatic updates from service libraries and the server operating system must be implemented. A static code scan should be implemented to ensure a secure development.

Besides of security measures, the functional correctness is important in order to prevent customers dissatisfaction. A unit test coverage of 70% and a basic coverage of every user story by integrated system tests ensure the functional correctness and prevent regressions.

Another quality measure to prevent customer dissatisfaction, an availability of 95% should be archived.

The service is expected to serve thousands of customer in the future. Therefore, the flexibility in terms of scalability and replaceability is important. Nevertheless, the first version only targets around 100 user with low usage rates. A concept for manual scaling and server replacement must exist.

The technical requirements ensure a fast and efficient development process and build the base for the feature development. In alignment with a lean development process, a continuous integration and deployment pipeline is required for a continuously flow of new items to the user. Next the backend server must be stetted up as a proxy to the OpenAI assistant API, but also to implement additional functionalities.

Technical requirements are maintained as tasks. Quality requirements are part of every activity (e.g. the functional correctness) or also maintained as task (e.g. the secure server task).

Every non-functional task is created via a Github Template an dcontains a basic description and acceptantce criteria.

[TASK]: CICD pipeline for server deployment **task**
#52 - by djochim was closed last week

[TASK]: Enable server monitoring via grafana **task**
#16 - by djochim was closed 4 hours ago

Secure cloudflare to server connection **task**
#14 - by djochim was closed on Feb 3

Connect custom domain **task**
#12 - by djochim was closed on Jan 29

Secure server **task**
#11 - by djochim was closed on Feb 15

Initial setup **task**
#10 - by djochim was closed on Jan 27


Add deployment pipeline for automatic backend deployments **task**
#3 - by djochim was closed 2 weeks ago - v1

Initialize Kotlin MMP project **task**
#2 - by djochim was closed last month - v1

[TASK]: CICD pipeline for server deployment #52

Edit New issue

Open Parent: [EPIC]: Setup cloud server #66

 djochim opened 2 weeks ago - edited by djochim

Edits

...


Description


To ensure seamless and automated deployment of server updates, a Continuous Integration and Continuous Deployment (CI/CD) pipeline should be implemented. This pipeline will automate the process of building, testing, and deploying new versions of the server whenever changes are merged into the main branch. By doing so, the development workflow becomes more efficient, reducing manual effort and minimizing deployment errors.

Acceptance Criteria

- A CI/CD pipeline is set up to automatically trigger upon merging changes into the main branch
- The pipeline includes build, test, and deployment stages to ensure quality and stability
- Successful builds are deployed to the production environment automatically

Create sub-issue

 djochim added **task** 2 weeks ago

 djochim moved this to In progress in SE Software project 2 weeks ago

Assignees

No one - Assign yourself

Labels

task

Projects

SE Software project

Status Done

Milestone

v1

Due by April 30, 2025, 23% complete

Relationships

Parent issue

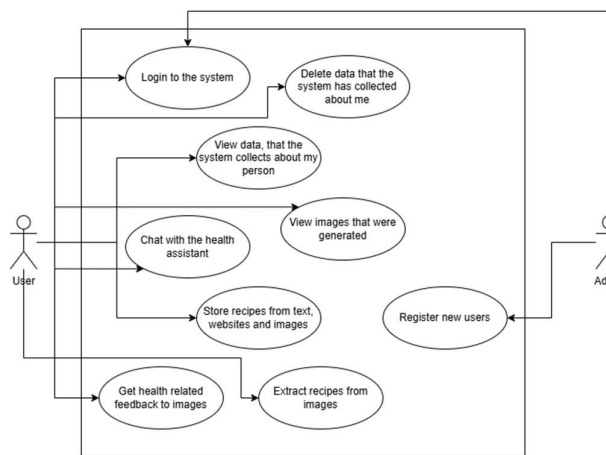
[EPIC]: Setup cloud server

djochim/cs_se_project#1

Functional

For the first milestone the health coach has a very limited scope. First of all, the basic communication will be enabled between the Assistant and the user. This communication is based on text messages and the user can also send images.

Furthermore, there will be a first toll implementation that allows the user to store recipes that are generated or extracted from text, images or websites. The use-Case-Diagram shows the stories for users and the few requirements for admins.



- 🕒 [STORY]: Users data deletion story
 #51 · djochim opened 2 weeks ago

- 🕒 [STORY]: Users can retrieve their user data automatically story
 #50 · djochim opened 2 weeks ago

- 🕒 [STORY]: User Login story
 #49 · djochim opened 2 weeks ago

- 🕒 [STORY]: Admin user registration story
 #48 · djochim opened 2 weeks ago

- 🕒 [STORY]: Sending images story
 #41 · djochim opened 2 weeks ago · 🔄 v1

- 🕒 [STORY]: List of generated recipes story
 #7 · djochim opened on Jan 27 · 🔄 v1

- 🕒 [STORY]: Assistant Tool: Recipe store story
 #6 · djochim opened on Jan 26 · 🔄 v1

- 🕒 [STORY]: Basic Nutritional assistant chat story
 #5 · djochim opened on Jan 26 · 🔄 v1

- 🕒 [STORY]: Authentication story
 #4 · djochim opened on Jan 26 · 🔄 v1

Feature enhancements are tracked via User Stories. User stories are created via a Template in Github and every user story contains a general description, acceptance criteria and the “As a [user/person] I want [action] so that [benefit]” phrase. Every user story is connected to a parent epic, which is grouping several related user stories. User stories can be released independently, and every story is adding a benefit for the user to the app.

Based on the status of a user story, the story is also prioritized and estimated. The priority is categorized in P0 (highest), P1 and P2 (lowest). For the estimation two values are used. The size of an item specifies the implementation effort that is required to implement something and is a good indicator how long it will take to implement an item. In contrast, the estimate number is more a definition how complex an item will be. Very complex items might require some spike to be implemented or further refinement / simplifications.

In addition, every user story is assigned to a milestone. In case of this project, every story is connected to milestone v1.

[STORY]: Basic Nutritional assistant chat #5

[Edit](#)
[New issue](#)

🕒 Open
 🔗 Parent: [EPIC]: Nutritional assistant chat

[👤](#)

djochim opened on Jan 26 · edited by djochim

Description

The chat feature is the core functionality of the nutritional assistant app, enabling users to receive personalized, real-time health and nutrition guidance through an intuitive conversational interface. Users can ask questions about diet, meal planning, exercise, hydration, and overall wellness, and the assistant will provide clear, accurate, and science-backed responses.

The chat is designed to be context-aware, allowing users to ask follow-up questions within a session for a smooth and natural interaction. If the assistant cannot answer a query, it will guide the user to relevant resources or alternative ways to get information.

With a user-friendly interface, quick response times, and adaptive learning capabilities, the chat ensures a seamless experience, helping users stay informed and motivated on their health journey.

As a
user

I want
chat with the nutritional assistant

so that
I can receive instant and accurate health advice tailored to my needs

Acceptance Criteria

- The user can send messages to the assistant via a chat interface
- The assistant responds to user inquiries about health, nutrition, and wellness
- The chat maintains context within a session, allowing follow-up questions
- The chat interface is intuitive, responsive, and accessible
- Error handling is in place for unrecognized queries, with a fallback response or guidance

Create sub-issue
👤

Assignees

No one · [Assign yourself](#)

Labels

Story

Projects

SE Software project
Status Refined

Priority P1

Size M

Estimate 5

Start date No date

End date No date

Milestone

🏠 **v1**
Due by April 30, 2025, 25% complete

Relationships

Parent issue

🔗 [EPIC]: Nutritional assistant chat
djochim/cs_se_project#40

Development

Create a branch for this issue or link a pull request.

Glossary

Chat Interaction	The communication between the user and the assistant via a Chat interface
Nutritional assistant	The assistant that provides dietary recommendations to the user
Recipe	A plan for cooking a meal based on ingredients and steps

System Design

The application will consist of an Android and iOS app based on the Kotlin Multi-Platform project. There the server, Android App and iOS APP is developed with Kotlin and all three are also able to share some of the code parts with each other. The backend server uses Ktor as a server framework and connects to a MongoDB to store required data.

The communication between the apps and the backend is based on the compact and efficient serialization format protobuf. Every request pass through Cloudflare and Nginx for caching and security optimizations.

Cloudflare acts as a reverse proxy and web application firewall. It protects the server against DDoS attacks and blocks traffic from known bot sites. Cloudflare is also used as the provider for the domain certificate and server certificates. In addition, Cloudflare can be used to cache static content like recipes or nutritional information of products.

The backend will be deployed on a Hetzner Cloud Server as Docker containers.

Nginx is an additional reverse proxy inside the server. It acts as a single entry point, only allowing traffic over the https port 433 and accepts only traffic that originates from Cloudflare.

The backend server implements the main business logic and handles the requests of users. It communicates with the public API of OpenAI for the implementation of the assistant functionality.

Authentication requests are handled by a separate server, which is not developed as part of this assignment.

MongoDB is used as the database, and it is used to store the chat histories, user preferences and goals and e.g. meal plans or recipes.

It is important to have a reliant infrastructure setup and therefore the server is monitored using Grafana. For this integration a docker container with Grafana alloy is deployed inside of the server and forwards required monitoring information to the Grafana Cloud instance.

